Optimizations for real-time implementation of H264/AVC video encoder on DSP processor

Nejmeddine Bahri, Imen Werda, Thierry Grandpierre, Mohamed Ali Ben Ayed, Nouri Masmoudi, Mohamed Akil

Optimization for Real-Time Implementation of Baseline H264/AVC Video Encoder on DSP Processor

N.Bahri, I.Werda, M.Ben Ayed, N.Masmoudi
National School of Engineers, LETI Laboratory
University of Sfax, Tunisia
nejmeddine.bahri@gamil.com

T.Grandpierre, M.Akil
ESIEE Engineering, LIGM laboratory
PARIS-EST University, France
t.grandpierre@esiee.fr

***Abstract*—Real-time H.264/AVC high definition video encoding represents a challenging workload to most existing programmable processors. Moreover, the new technologies of programmable processors such as GPU and multicore DSP offer a very promising solution to overcome these constraints. In this paper, an optimized implementation of H264/AVC video encoder on a single core among the six cores of TMS320C6472 DSP for CIF (352x288) resolution is presented in order to move afterwards to a multicore HD implementation. Algorithmic optimization was applied to the intra prediction module to reduce the computational time. Furthermore, based on the DSP architectural features, various structural and hardware optimizations were adopted to minimize external memory access. The parallelism between CPU processing and data transfers was fully exploited using an EDMA controller. A second core is reserved to perform real time frame capture from a digital camera to the DSP memory using the Ethernet protocol. Experimental results of our structural and hardware optimizations on a single core running at 700 MHz for CIF resolution improve the encoding speed by 33.95%. The proposed fast intra prediction algorithm can save 58% of the intra prediction computational time and accelerate the encoding speed by 13.58% without inducing any PSNR degradation or bit-rate increase. All these optimizations lead to a total reduction in execution time by up to 42.91% satisfying the real-time encoding 25 f/s and allowing the possibility to reach real time implementation for HD resolutions when exploiting multicore features.**

Keywords—H264/AVC encoder, TMS320C6472 DSP, algorithmic and structural optimizations, EDMA, real time.

I. INTRODUCTION

H.264/AVC [1] is a video encoder standard. It achieves better video coding efficiency by saving up to 50% of bit-rate as compared to previous standards and maintaining the same visual quality. However, this efficiency is accompanied by a high computational complexity due to many new modules included in this standard such as intra and inter predictions designed to reduce spatial and temporal redundancies respectively [1]. When moving to HD resolutions, the encoding time is drastically increased. This requires high-performance processing capability to satisfy the real time constraint 25 f/s. Many investigations have been proposed to reduce the computational complexity of the H264/AVC encoder. Algorithmic optimizations are applied in order to reduce the computational time without affecting the rate-distortion [2],…,[15]. Other works implement specific modules on hardware accelerators such as FPGA platforms while keeping the algorithmic part of the encoder on another processor [16],…,[20]. This unfortunately induces extra-communication latencies among the different components of the platform. Finally, some propositions have profited of the new multicore technologies of embedded processors to achieve parallel processing by partitioning different tasks on different units [21],…,[23]. Actually, most of the existing real-time H.264/AVC encoders are implemented on DSP platforms since they offer software flexibility that is important to allow upgradability but also relatively low software development cost and time-to-market reduction [24],…,[33]. The counter part is that their computing power doesn't allow encoding real-time HD resolution without internal fixed IP that can't be upgraded to follow latest protocol enhancements. In order to reach real time HD video encoding and achieve a better encoding efficiency in terms of video quality and bit-rate comparing to H264/AVC existing DSP implementations on the market, we propose to exploit the merits of multicore DSP such as the Texas Instruments' DSP TMS320C6472 which is based on 6 cores. In this paper, we present an optimized implementation of the baseline H264/AVC encoder on a single CPU core among the six cores of TMS320C6472 DSP for CIF (352x288) resolution. Based on this work, we hope to move afterwards to a multicore HD implementation. The rest of our paper is structured as follows: next section presents an overview on H264/AVC encoder structure. Section three details the related works on H264/AVC encoder optimizations. The internal architecture of our platform is described in section four. Section five highlights our proposed optimizations. Experimental results and discussion are presented in section six. Finally, section seven concludes this paper and present perspectives of HD encoding on multicore.

II. OVERVIEW OF H264/AVC ENCODER STRUCTURE

The H.264/AVC video coding structure as shown on "*Fig. 1*" is a hybrid of temporal and spatial predictions. The principle of coding a video sequence is detailed as follows: for the baseline profile, there are two frame types: I frames, where only the intra prediction module is applied and P frames where both intra and inter predictions are performed. The frame is

This work is a cooperation between the National School of Engineers of Sfax Tunisia and ESIEE Engineering PARIS. It is sponsored by the French ministries of Foreign Affairs and the Higher Education and Research and the Tunisian ministry of Higher Education and Scientific Research in the context of Hubert Curien Partnership (PHC UTIQUE) under the CMCU project number 12G1108.

divided into macroblocks (MB) of 16x16 pixels. Each MB undergoes the two prediction types:

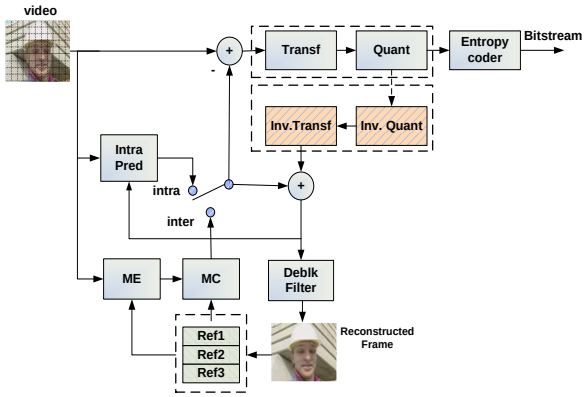


Fig. 1. H264/AVC video encoder structure

Intra-prediction: it is dedicated to remove the spatial redundancies within the image and includes two different block-size predictions:

- Intra4×4: it is performed for each 4x4 block and containing nine directional prediction modes as shown on *"Fig. 2"*.
- Intra16×16: it is applied for each 16x16 MB and including four directional prediction modes as illustrated on *"Fig. 3"*.

To perform intra prediction, the current MB needs the TOP, the LEFT and the TOP LEFT neighboring pixels of its neighboring MBs already encoded and reconstructed as illustrated on *"Fig. 4"*.

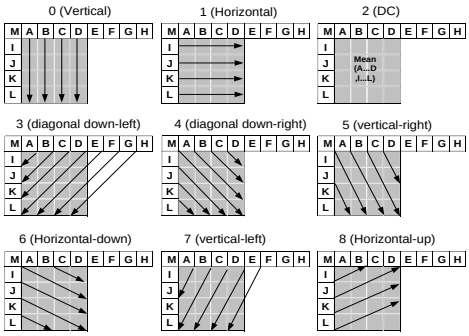


Fig. 2. Intra4x4 prediction modes

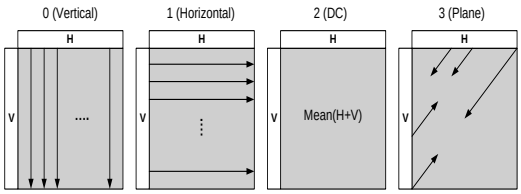


Fig. 3. Intra16x16 prediction modes

Inter-prediction: It is used to reduce temporal redundancies in the video. It consists of determining the motion vector of the current MB in frame i with respect to its position in multiple reference frames i-n. n represents the time difference between the actual encoded frame and the reference frame within the display order. The search of the motion vector is limited to a specific area called "search window" as presented on *"Fig. 5"*.

Fig. 4. Data dependencies for intra prediction

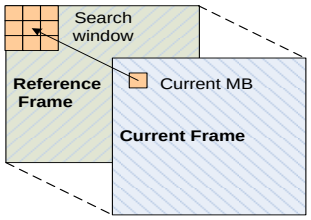


Fig. 5. Data dependencies for inter prediction

For inter prediction, there are seven different block sizes, as shown on *"Fig. 6"*, namely 16×16, 8×16, 16×8, P8x8 {8×8, 8×4, 4×8 and 4×4} in order to reach more motion vector precision. Many rate distortion calculations are performed to determine the best predicted MB between all the intra and inter prediction modes using the following formula:

$$Cost_{mode} = Distortion\ (MB) + \lambda_{mode}\ x\ Rate\ (MB) \qquad (1)$$

The Distortion is determined by means of the most commonly metric which is the Sum of Absolute Difference (SAD) computed between the current MB and the predicted MB. The mode that minimizes the cost is chosen as the best prediction mode.

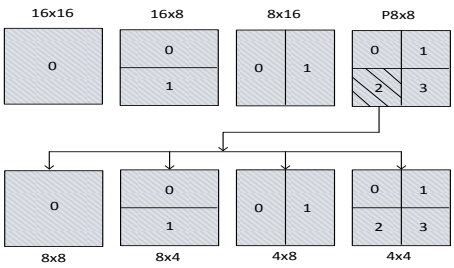


Fig. 6. Inter prediction modes

A DCT-like integer transform and quantification are applied to the residual MB which is the difference between the current and the best predicted MB. Transform coefficients are quantized and entropy coded to form the bit-stream to be transmitted or stored in a file. The decoding chain composed by the inverse quantification and inverse transform is used to reconstruct the MB needed for the prediction of the subsequent MBs. The reconstructed frame is filtered using a de-blocking filter to remove artifacts. This filter requires the use of 4 rows of pixels for the TOP and 4 columns of pixels for the LEFT neighboring MBs as shown on *"Fig. 7"*.

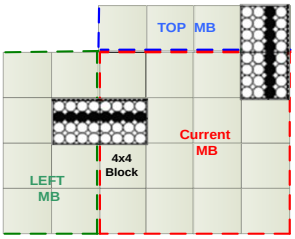


Fig. 7. Data dependencies for de-blocking filter

III. RELATED WORKS

A. Software solutions

The H264/AVC modules profiling shows that inter and intra predictions represent the lion's share of the encoder computation time. Using a fast motion estimation approaches [24], computational complexity of inter prediction module is drastically reduced. As a result, the required time for checking intra prediction modes becomes relatively important compared to that of the inter prediction module. Therefore, a fast intra mode decision algorithm is required. This complexity is due essentially to the Rate Distortion (RD) computations performed for each mode in order to select the best one. Many solutions and techniques have been proposed such as:

- Reducing the number of candidate modes instead of performing all the RD calculations of intra4x4 and 16x16 prediction modes [2],…,[6].
- Comparing boundaries variance and/or MB variance to a threshold in order to achieve early termination of cost computation [7],…,[10].
- Extracting the direction MB histogram to have an idea of the MB direction in order to reduce the candidate modes [11],…,[15].

B. Hardware/software solutions

Many hardware/software co-design architectures for H.264/AVC encoder are proposed in order to take advantage of hardware parallelism. Based on H264/AVC encoder profiling, a partitioning of the H264 modules is determined. In fact, the most time-consuming and systematic modules are accelerated using hardware IPs implemented on Field Programmable Gate Array (FPGA) platforms [16],…,[20]. The rest of the modules will be performed on another processor. Then the communication between the processors and the hardware block is examined.

C. DSP based solutions

Software flexibility, time-to-market reduction, low power consumption and low cost make DSPs an attractive solution for high performance applications and embedded systems implementations. Motivated by these merits and encouraged by the great evolution of DSP architectures, several proposals have taken advantage of DSP features (high processing frequency, optimized architecture) in order to achieve real time video encoding [24],…,[33].

D. Analysis of previous works

Concerning the algorithmic optimizations cited above for the intra prediction module, even if some of these algorithms were able to reduce the intra prediction computation time, they still have some drawbacks. First, they increase the H264/AVC complexity computation due to the pre-calculations required for computing the edge direction and boundary variance. Moreover, methods reducing the number of intra candidate modes affect only the RD calculations whereas intra prediction also includes other modules such as computing residual block, transform, quantification which are performed to prepare the neighboring pixels for the next block. Finally, it is crucial to consider that these approaches induce a bit-rate increase when the rate control option is turned off and lead to a serious degradation of PSNR quality. On the other hand, inflexibility and the inherent long time to market are the most important drawbacks of FPGA implementations. Thus, to implement the whole H264/AVC encoder on an FPGA platform, a huge FPGA surface and a lot of design and compilation time with tremendous VHDL expertise are required. Finally, the low hardwired block frequency makes it difficult to attain real time encoding especially for HD resolutions. Regarding the DSP based solutions; the existing mono-core DSP encoder implementations still cannot meet real-time constraints particularly for HD resolutions. In fact, it is only with large memory-size and high clock frequency platforms that real-time H.264/AVC encoding is possible.

IV. DSP PLATFORM DESCRIPTION

Our choice for using a DSP platform was motivated by its merits described above. To reach real time encoding for HD quality as our final goal, we decide to work on multicore DSP which is the renowned TMS320C6472 DSP [34]. It belongs to the latest generation of multicore DSPs made by Texas Instrument. Low power consumption and a competitive price tag make the TMS320C6472 DSP ideal for high-performance applications and suitable for many embedded implementations. As presented on *"Fig. 8"*, six C64x + DSP cores, 4.8 M-Byte (MB) of memory on chip, very long instruction word (VLIW) architecture, Single Instruction Multiple Data (SIMD) instruction set and a frequency of 700 MHz for each core are combined to deliver 33600 MIPS performance.

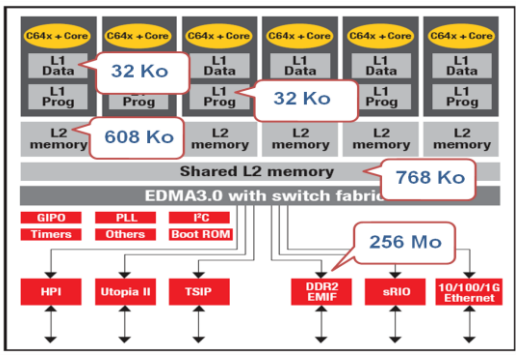


Fig. 8. Internal architecture of TMS320C6472 DSP

Each C64x+ core integrates a large amount of on-chip memory organized as a two-level memory system. The level-1 (L1) program and data memories on this C64x+ core are 32 K-Byte (KB) each. This memory can be configured as mapped RAM, cache, or some combination of the two. The level 2 (L2) memory is shared between program and data space and is 608 KB in size. L2 memory can also be configured as mapped

RAM, cache, or some combination of the two. In addition to L1 and L2 memory dedicated to each core, the six cores also share 768 KB of L2 shared memory. Shared L2 memory is managed by a separate controller and can be configured as either program or data memory. This large quantity of on-chip memory can eliminate access to external DDR2 memory, therefore reducing the power dissipation and accelerating the execution of algorithms since internal memory is faster than external memory. Performance is also enhanced by the EDMA controller able to manage memory transfers independently from the CPU. Therefore, no additional overhead is caused when large data blocks are moved between internal and external memory. TMS320C6472 DSP supports different communication peripherals as Gigabit Ethernet for IP networks, UTOPIA 2 for telecommunications and Serial RapidIO for DSP-to-DSP communications.

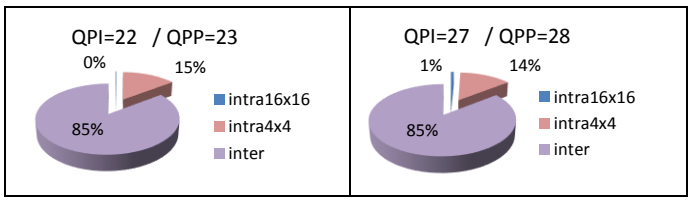
V. PROPOSED OPTIMIZATIONS

To take advantages of multicore technology and the potential parallelism presented in the H264 standard, we must as a first step, elaborate an optimized H264/AVC architecture on a single core DSP even for a low resolution such as CIF format and then we should be able to move to HD multicore implementation. The video frame is captured from a camera and converted into YUV 4:2:0 format adequate for the baseline profile. Then, the RAW stream is transferred to the DSP memory in order to be processed by the DSP core to obtain the compressed bit-stream subsequently transferred to a video decoder or stored in a file. To achieve real time encoding, several optimizations are proposed and implemented hereunder on a unique DSP core.

A. Algorithmic optimization

To overcome the drawbacks of previously proposed algorithms for the intra prediction module, a fast intra prediction approach is presented. When adopting this approach, pre-calculations are not required to achieve early termination mode decision. The aim of the proposed approach is to reduce the whole intra prediction complexity and not only the RD calculations as adopted by most of the previous algorithms. The best way to achieve this proposal is choosing the appropriate condition to decide either performing or skipping the whole intra4x4 or intra16x16 module. Our proposed scheme is a result of several analyses that we performed on several CIF sequences (Akiyo, Foreman, Container, News, Tb420, mobile, Bus, ice…etc). These analyses show that:

- For P frames, the inter mode is the most selected mode compared to intra16x16 and intra4x4, as presented on *"Fig. 9"* which shows the mode decision percentages for CIF video sequences with different quantification parameters (QPI, QPP). As a result, when a fast intra mode decision is performed in P frames, no important degradation in visual quality is noticed.



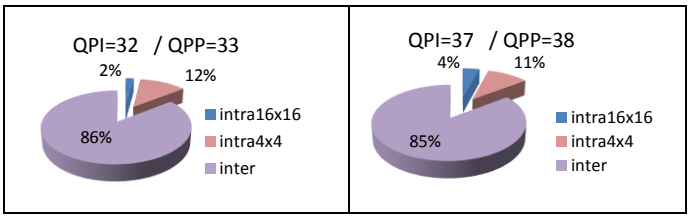


Fig. 9. Mode decision percentages for CIF video sequences

- 16x16, 16x8 and 8x16 inter prediction block sizes are generally used for backgrounds and stationary blocks whereas P8x8 block sizes are used in detailed and fast motion areas as illustrated by *"Fig. 10"*.

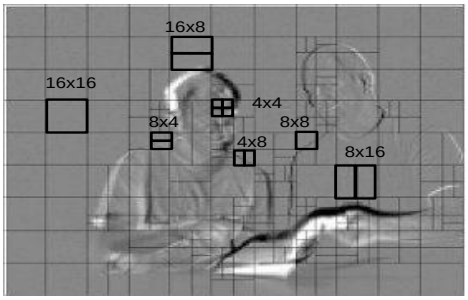


Fig. 10. MB regions for inter prediction modes

- Intra16x16 is generally used for backgrounds, stationary and homogenous blocks characterized by a tiny luminance change, whereas intra4x4 is used for high luminance texture areas as shown on *"Fig. 11"*.

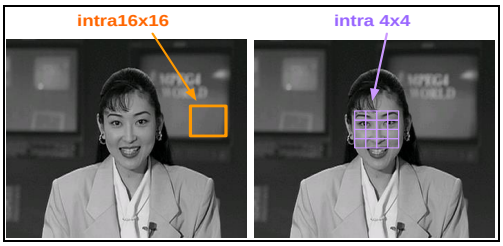


Fig. 11. MB regions for intra prediction modes

Considering these observations, we can affirm that there is a high correlation between intra and inter prediction modes. So, a fast intra mode decision based on inter prediction modes is proposed. This is depicted in the flowchart of *"Fig. 12"*:

For I Frames, intra mode decision is not changed: the full intra prediction process is performed and all intra16x16 and intra4x4 modes are checked.

For P Frames, the intra mode decision is linked to the best inter prediction mode as follows:

- If the best inter mode is P8x8, which means that 8x8, 8x4, 4x8 or 4x4 block size is selected, intra4x4 is performed and the whole intra16x16 is skipped because the MB is considered not homogenous and characterized by a detailed texture.
- Otherwise, intra16x16 is performed and the whole intra4x4 is skipped because the MB is considered a homogenous smooth area.

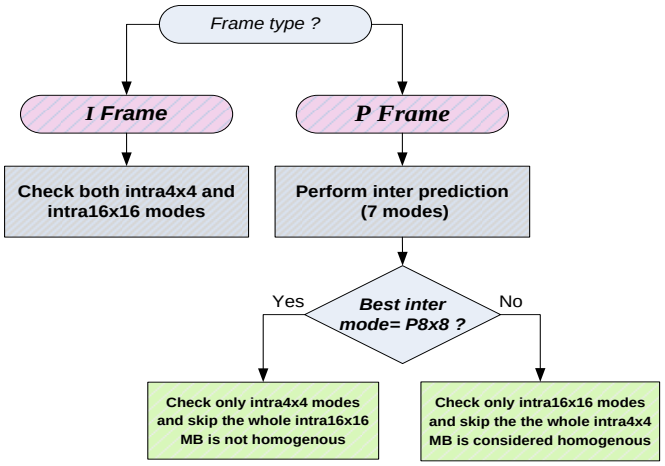


Fig. 12. Flowchart of the proposed algorithm

For performance comparison, our proposed approach will be evaluated according to three criteria:

- ΔPSNR: It is the PSNR difference between the proposed and the reference algorithms.
- ΔBits (%): It represents the percentage increase in bit-rate for the implemented scheme compared to the reference software.
- ΔTime (%): This criterion is defined as the percentage of time saving for intra prediction module when performing the proposed scheme.

The above three criteria are detailed by the following equations:

$$Bits(\%) = \frac{\text{Bitrate(proposed)} - \text{Bitrate(reference)}}{\text{Bitrate(reference)}} \text{x100} \quad (2)$$

$$\Delta PSNR(db) = PSNR(proposed) - PSNR(reference) \quad (3)$$

$$\Delta Time(\%) = \frac{Time(proposed) - Time(reference)}{Time(reference)} x100 \quad (4)$$

Table I shows the performance of our scheme realized on 5 CIF sequences: Akiyo, Foreman, News, Paris, and Container.

TABLE I. PERFORMANCE EVALUATIONS OF THE PROPOSED ALGORITHM

Sequence	ΔPSNR (dB)	ΔBits (%)	ΔTime (%)
Akiyo	-0.00	+0.00	58.7
Foreman	-0,02	+0,14	57.2
News	-0.00	+0,17	59.0
Paris	-0.01	-0,003	55.6
Container	-0.00	+0,01	58.4

As presented in this table, we can observe that our proposed algorithm does not affect the PSNR quality. It provides a slight increase in bit-rate by 0.07% in average. Finally, our approach can save up to 58% of intra prediction computation time.

B. *Data structure optimizations*

It consists of designing and optimizing data structures that enable the processing of input video into H264 encoder modules. This structure should efficiently exploit the DSP core architecture and especially the internal memory which is faster than external SDRAM memory. Each core of TMS320C6472 DSP has a 608 KB internal memory LL2RAM shared between program and data. Preferably and to the extent possible, we should load both program and data within the LL2RAM. Two implementation variants are proposed.

1) « MB level » implementation

This implementation represents the basic data structure processing in H264/AVC standard, based on encoding a MB followed by another until we finish the entire frame MBs. The principle of this first proposed architecture is detailed as follows: the program size is around 120 KB and is loaded into the internal memory LL2RAM (608 KB). As a result, 488 KB of memory space remains free. For YUV 4:2:0 baseline profile format (for each 4 pixels, we have 4 luminance Y, 1 chrominance U, and 1 chrominance V components). Consequently, for a CIF resolution, each frame requires 148.5KB (=352x288x1.5). If we consider the reference frame, we have to extend its 4 sides by a MB (16 pixels) needed for the window search in order to perform the motion estimation. In our implementation, only one reference frame is used, so a total of 180KB (= (352+32) x (288+32) x 1.5) are required. In addition, 64 KB are reserved for the compressed output bit-stream. All these data are stored in the external memory given their large sizes. To avoid working directly with external memory, some data are copied into the internal memory such as current MB, window search and reconstructed MB for the 3 YUV components. Quantization and transform matrixes, predicted MBs, SAD matrix, neighboring pixels buffers are also loaded into the LL2RAM in order to accelerate the data processing and minimize external memory access. The total size of data loaded into LL2RAM is 28.24 KB, so 459.76 KB of internal memory are not allocated so still free for future enhancements. The steps for encoding a luminance MB are detailed on *"Fig. 13"*.

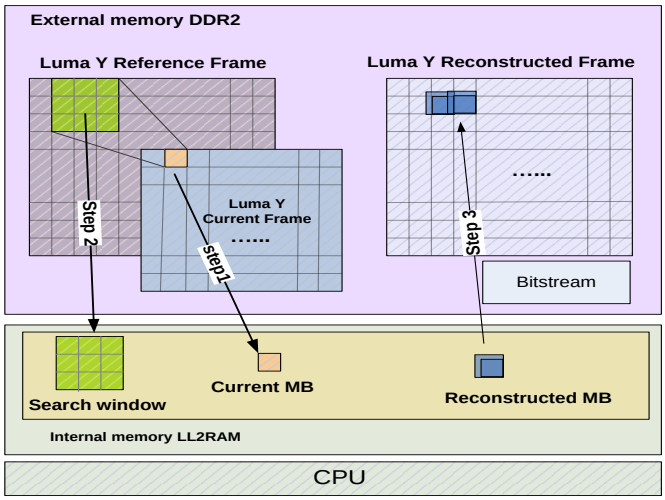


Fig. 13. « MB level » implementation

The DSP core loads the current MB (16x16) and the search window (48x48) respectively from the current and the reference frames from the external to internal memory. The data process is now totally performed by the DSP core without

external memory accesses. The reconstructed MB (20x20), extended by 4 pixels at the left and the top needed in the MB filtering, is transferred from the local memory into the external memory at the reconstructed frame. This process is repeated until the completion of the entire current frame MBs. The advantages of this architecture are essentially its adaptability to any DSP even if it does not have a large internal memory and small internal memory requirement even for HD resolution (55.54 KB for 720p resolution). We notice that it is practically independent from the video resolution. The major drawbacks of this architecture are the multiple accesses to the external memory for each loading of a current MB, reading of the search window and saving of the reconstructed MB in the reconstructed frame. It also needs to save the left and top neighboring pixels used in the prediction and filtering of the next MBs after each MB processing.

2) « One MBs row level» implementation

To avoid the first architecture's drawbacks, a second implementation is proposed. The principle of this implementation as shown on *"Fig. 14"* consists of loading one MBs row (16 x frame_width) from the current frame and 3 MBs rows (48 x (16+ frame_width +16)) for the search window from the reference frame to the appropriate buffers created in the internal memory. The DSP core encodes the whole current MBs row without external memory access. Then, the reconstructed MBs row (20 x (16+ frame_width +16)) is transferred from the LL2RAM to the SDRAM memory in the reference frame. Thus, it is not necessary to create another memory buffer for the reconstructed frame. We can exploit the reference frame to store the reconstructed MBs row, since the overwritten data are not useful (they are already copied into the 3 MBs rows of the search window). Passing to the second current MBs row, it is not necessary to load 3 MBs rows for the search window from the reference frame, just shift up the last two MBs rows of the search window in the internal memory and bring the third from the fourth line of the reference image as presented on *"Fig. 15"*.

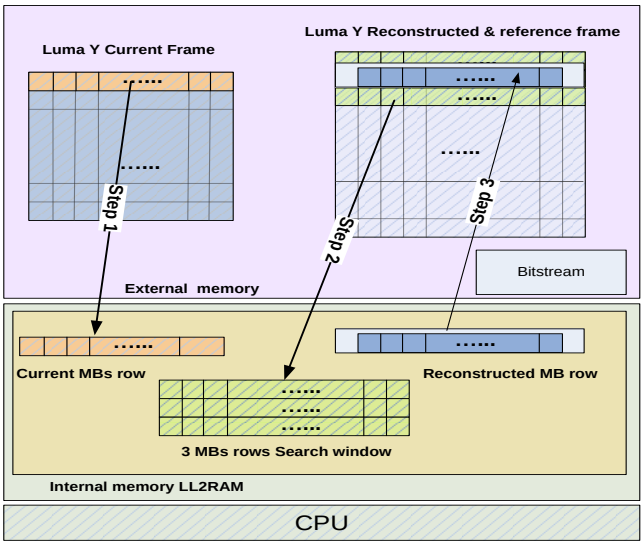


Fig. 14. « one MBs row level » implementation

The required data size in the internal memory for the CIF resolution for this second proposition is 68.56 KB instead of 28 KB for the first architecture. However, it outstandingly reduces the access to external memory. Finally, instead of performing multiple accesses to the external memory for loading and storing data for the "MB level" architecture which are in a total of 22 (number of MBs per row for CIF resolution); we only perform one access for "one MBs row level" architecture. In addition, when proceeding at the MBs row level, all left boundaries required in the next MB prediction and filtering are already available in the memory, so the left neighboring pixels backup is removed. Moreover, this reduces the backup of TOP boundaries, since we require storing the top boundaries only one time after finishing the processing of the whole MBs row whereas, the "MB level" implementation needs to store the top neighboring pixels after processing each current MB.

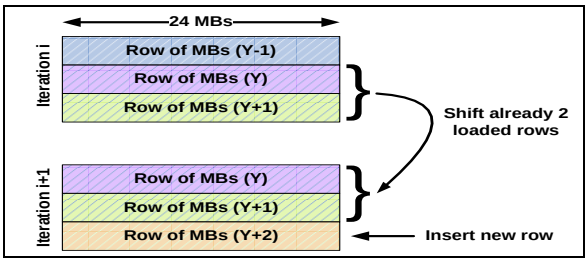


Fig. 15. Reference search window load

C. Hardware optimizations

1) « ping pong MBs row level» implementation: EDMA based solution

To reduce the H264/AVC encoding latency, we propose a bus interleaved architecture. A ping-pong buffer is configured to support the current and reconstructed MBs transfers, ensuring better parallelism efficiency between data transfers and CPU processing of the next MBs row. The C6472 DSP includes an Enhanced Direct Memory Access controller (EDMA) [35] which handles data transfers programmed between two memory-mapped slave endpoints without direct CPU involvement. The parallelism between CPU processing and EDMA data transfers can significantly accelerate the DSP processing. Ping-pong buffers are configured to support the current and reconstructed MBs transfers.

Three execution phases are performed as follows:

- Phase 1: While the CPU encodes the current MBs rows composed of one luminance (Y) and two chrominance (UV) components on the "ping" buffer, three DMA requests start loading the three components (YUV) respectively from the following MBs rows located at the current frame buffers in external memory into the correspondent "pong" buffers in internal memory. Consequently, with this technique, we can save the transfer time of the current MBs rows.

- Phase 2: the filtering module is performed after the CPU terminates encoding all the MBs in the current row. Thereby, we can exploit this processing order to parallelize MBs row filtering with preparing the three MBs rows of the search windows for motion estimation of the next current MBs row (pong buffer). The CPU performs the filtering module of the whole "ping" reconstructed MBs rows (luma and chroma

components), in parallel, the EDMA controller shifts up the last two rows of the search window in the internal memory and load the third from the correspondent row of the reference image located at the external memory. This phase permits to reduce the transfer time of the search window.

- Phase 3: at the end of the "ping" reconstructed MBs rows, the CPU moves to process the content of the "pong" buffer that becomes the new current MBs rows. At the same time, three EDMA channels are triggered to save the "ping" reconstructed MBs rows for the three components, luma and two chromas, from the internal to the external memory. Then, three other EDMA channels start loading the next current MBs rows on the "ping" buffer, and so on. The advantage of these steps is to reduce the encoding latency: the CPU doesn't need to wait for storing the reconstructed MBs rows to move to processing the next current MBs rows as described on *"Fig. 16"*.

The whole data amount in internal memory for this architecture is 88 KB only for CIF resolution. Therefore, 400 KB are still not utilized; it will be interesting for higher resolution encoding.

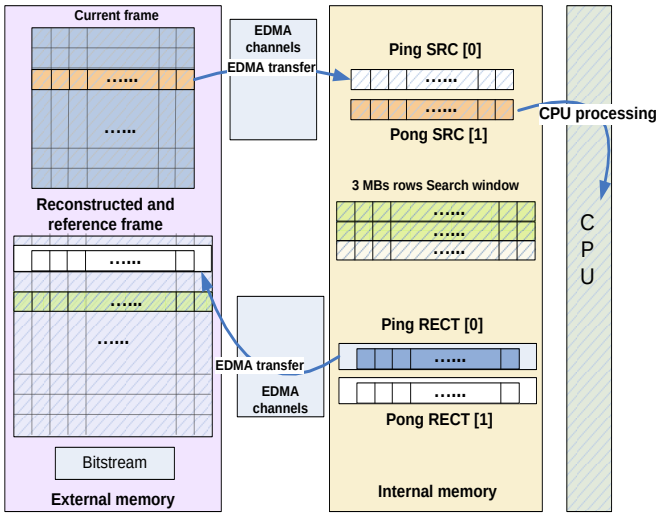


Fig. 16. Phase 3 of the « ping pong MBs row level» implementation

2) Cache activation

Each local L2 memory of DSP cores can be configured as all SRAM or as part 4-way set-associative cache (32 KB, 64 KB, 128 KB or 256 KB). As we still have 400 KB of free space in L2 memory, 256 KB are configured as cache memory in order to speed up the data processing, reduce CPU access time (read or write) to its data, and minimize the probability of cache misses. To activate the local L2 cache, the chip support library (CSL) API "CACHE_setL2Size" is used [36]. Also, we must enable caching external address ranges by setting the Memory Attribute Registers (MARs) using the CSL API command "CACHE_enableCaching".

3) Real time video encoding demo

For real time experimentation, the video input for the H264/AVC encoder needs to capture frames from camera at the rate of 25 f/s. For 4:2:0 video format, we need a transmission channel bandwidth equal to 29 Mbps for CIF resolution ((352 x 288 x 1.5) x 8bits x 25 f/s) and 263.67 Mbps for HD 720p resolution ((1280 x 720 x 1.5) x 8bits x 25 f/s). To perform a real time video encoding demo, we used a USB webcam connected to a Personal Computer (PC) which sends the raw YCrCb pixels to the DSP board using the Gigabit Ethernet link of the C6472 DSP. A TCP socket client (PC)-server (DSP) communication is established [37]. Texas Instrument Network Developers Kit (NDK) is used on the DSP side [38]. It provides a TCP/IP stack, network applications, and EMAC device drivers that work with DSP/BIOS the real time OS from TI. The phase of sending and receiving data between the client (PC) and the server (DSP) is presented on *"Fig. 17"* and detailed as follows: on the DSP, two cores are exploited. One is devoted to establish TCP/IP connection with the client (PC). It is engaged at first to receive the current frame sent by the client after camera capture and save it into the external memory. The second core will encode this current frame and save the bit-stream in the external memory to be thereafter sent by the first core to client (PC) in order to be stored in a file or decoded. A ping pong buffers for the current frame and the bit-stream are configured on the external memory in order to parallelize the current frame encoding by core 1 and reception of the next frame by core 0.

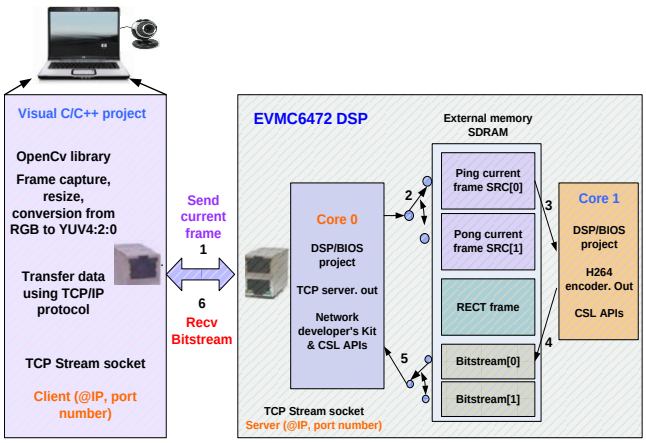


Fig. 17. Real time video encoding demo

The strategy of our implementation is described on *"Fig. 18"* and consists of the following steps:

- Establish connection between client (PC) and server core (0): create sockets, assign address to sockets (IP address, TCP port number) and wait for incoming connection by the server.
- On the client side, The PC captures frame from the camera or a file and converts it from RGB to YUV 4:2:0 format using the API openCv library [39].
- The client sends the captured frame to the server core (0) which will receive the stream socket and save it into the external memory in the ping buffer SRC[0]. Core (1) is in a wait state.
- Once the current frame reception is finished, core (0) sends an inter processor communication interruption

(IPC) to core (1) to trigger the encoding of the current frame SRC[0]. The generated bit-stream is stored into the ping buffer Bitstream[0]. At the same time, the client sends the next frame to core (0) which will receive this frame and store it into the pong buffer SRC[1]. Then, core (0) remains in a wait state until the reception of an IPC interruption from core (1) to indicate that the bit-stream is ready in the external memory after achieving the encoding of the ping buffer SRC[0].

- When receives an IPC interruption from core (1), core (0) sends an IPC to core (1), which is in a wait state, in order to trigger the encoding of the pong buffer SRC[1] and to ensure that the core (1) will not starting the processing before core (0) finish receiving the next current frame. Then, core (0) sends the ping buffer Bitstream [0] from external memory to the client (PC) and receive the next frame and save it into SRC[0]. At the same time, core (1) processes the pong buffer SRC[1] and saves the bit-stream into the pong buffer Bitstream [1] in order to avoid the overlap with the core (0).
- The work is then looped in a reverse order of current frame and bit-stream through ping pong buffers.

Considering this method, we can save the time of data transfers between the client and the server.

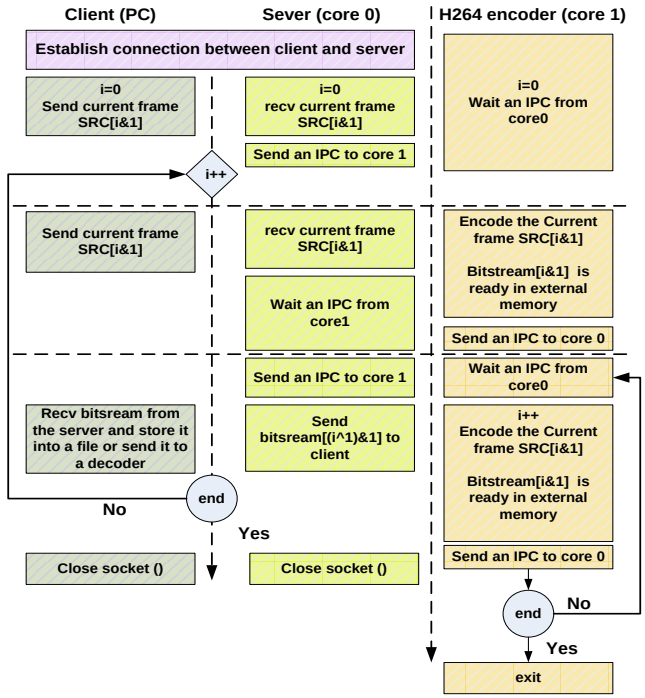


Fig. 18. The steps of performing real time demo video encoder

4) cache coherency

Multicore processing often leads to a cache coherency problem. This is due to the simultaneous access of two or more CPUs with a separate cache memory for each core to the same location in a shared memory. In general purpose multiprocessor, programmers don't have such problem because it is controlled automatically by complex hardware. But in our multicore DSP architecture, designers have to control it since there is no such automatic controller. In order to deal with cache coherency, the CSL library [36] from TI provides two API commands:

- CACHE_wbL2((void *)XmtBuf, bytecount, CACHE_WAIT) to write back the cached data from the cache memory to its location in the shared memory.
- CACHE_invL2((void *)RcvBuf, bytecount, CACHE_WAIT) to invalidate cache lines and oblige the CPU to read the data from its location in the shared memory.

In our case, when core (0) receives the current frame from the client, it should write back the cached data to external memory. In the other side, core (1) should invalidate the current frame addresses in the cache before starting the frame encoding. Also, when core (1) terminates the encoding frame, it should write back the bit-stream from cache to external memory in order to overcome the cache coherence with core (0) which will send the bit-stream from the external memory to the client (PC).

VI. EXPERIMENTAL RESULTS

The proposed architectures with different optimizations are implemented on a single core of the TMS320C6472 DSP running at 700 MHz. Experimental simulations are performed using CIF (352x288) video sequences with different characteristics: Foreman, Akiyo, News and Container. Simulation results are presented in Table II that shows the encoding speed of the different implemented architectures. The MB level implementation that was firstly presented, correspond to the basic encoding design and is consequently the slowest. It achieves 14.71 f/s which is very far from real-time. When moving to the second implementation "one MBs row level", the encoding speed is improved by 26.34% and reached 19.97 f/s. Then, parallelizing data transfers and CPU processing by exploiting EDMA controller and cache activation lead to an increase in speed by up to 10.33% allowing to achieve 22.27 f/s. Applying the proposed fast intra mode decision algorithm, the encoding speed is enhanced by 13.58%, surpassing the real-time constraint which is 25f/s without inducing any PSNR degradation or bit-rate increase as detailed in section 5.A. Finally we can note that the applied optimizations enabled us to achieve a gain of 42.91% starting with a slow video encoder running at 14.71 f/s before optimization and get a real-time video encoder running at 25.77 f/s. Our platform DSP using only a single core allows a real time video encoding for low resolution CIF (352x288).

Starting from these results, it is now possible to consider real-time for higher resolutions if we exploit the potential parallelism of the H264 standard and the merits of our multicore DSP. We have to explore the different partitioning methods ("GOP Level parallelism", "Frame Level parallelism" ...) and apply the most suitable for a multicore implementation. Also, the textural and morphological characteristics of the high-definition videos could be exploited to propose new algorithmic optimizations for the most H264/AVC complex modules as intra prediction and inter prediction.

TABLE II. PERFORMANCE EVALUATIONS OF THE PROPOSED IMPLEMENTATIONS

CIF sequence	MB level implementation	One MBs row level implementation	Ping Pong MBs row level implementation: EDMA	Ping Pong MBs row level implementation: EDMA solution + cache activated	Ping Pong MBs row level architecture: EDMA solution + cache activated + Fast intra mode decision algorithm
Foreman	14.73	19.96	21.48	22.41	25.07
Akiyo	14.83	20.19	21.71	22.40	25.98
News	14.73	20.02	21.49	22.29	26.47
Container	14.56	19.73	21.12	22.01	25.56
speed average (f/s)	**14.71**	**19.97**	**21.45**	**22.27**	**25.77**

VII. CONCLUSION

In this paper, an optimized implementation of the H264/AVC encoder on a single core of the multicore DSP TMS320C6472 was detailed. Algorithmic, structural, and hardware optimizations were proposed to achieve real-time coding 25 f/s on a single DSP core for CIF resolution (352x288). Our optimizations could save up to 42% of the total computation time and allows reaching 25.77 f/s in average as encoding speed for CIF resolution without visual quality degradation or bit-rate increasing. To switch to higher resolutions like SD and HD 720p, a multicore implementation on the 6 DSP cores of TMS320C6472 represents the next step that we should perform in order to attain real time encoding. Also we can benefit from the textural and morphological characteristics of HD resolutions to propose new algorithmic optimizations in order to accelerate the encoding speed. Finally, using the latest multicore DSP from Texas Instruments, the C66x DSP TMS320C6678 integrates eight cores each running at 1.25 GHz, is expected to achieve real-time encoding for HD 1080p resolution.

REFERENCES

[1] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, "Draft ITU-T Recommendation and Final Draft international Standard of Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC)", JVT-G050, 2003.

[2] Taeho Kim, Jechang Jeong "Fast Intra Mode Decision Using the Angle of the Pixel Differences along the Horizontal and Vertical Direction for H.264/AVC," Advances in Visual Computing, Lecture Notes in Computer Science, Springer, Volume 7432, 2012, pp 648-656.

[3] A. Elyousfi, A. Tamtaoui and E. Bouyakhf, "A New Fast Intra Prediction Mode Decision Algorithm for H.264/AVC Encoders," International Journal of Electrical and Electronics Engineering 4:1 2010.

[4] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu, and S. Wu, "Fast mode decision algorithm for intra prediction in H.264/AVC video coding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 7, pp. 813-822, July 2005.

[5] Chao-Chung Cheng and Tian-Sheuan Chang, "Fast Three Step Intra Prediction Algorithm for 4x4 blocks in H.264," Proc. IEEE Canadian Conference on Electrical and Computer Engineering, pp1981-1984, May 2003.

[6] Jun Sung Park and Hyo Jung Song, "Fast selective intra mode decision H.264/AVC," IEEE Consumer Communications and Networking Conference 2006.3rd, Vol.2, pp.1068-1072 Jan. 2006.

[7] Do Quan and Yo-Sung Ho, "Categorization for Fast Intra Prediction Mode Decision in H.264/AVC," IEEE Transactions on Consumer Electronics, Vol. 56, No. 2, May 2010.

[8] Yi-Hsin Huang, Tao-Sheng Ou, and Homer H. Chen, "Fast Decision of Block Size, Prediction Mode, and Intra Block for H.264 Intra Prediction," IEEE transactions on circuits and systems for video technology, Vol.20, No.8, august 2010.

[9] Mohammed Golam Sarwer and Q. M. Jonathan Wu, "Improved Intra Prediction of H.264/AVC," Effective Video Coding for Multimedia Applications, Sudhakar Radhakrishnan (Ed.), ISBN: 978-953-307-177-0, InTech (2011).

[10] Chi-Chou Kao, Yen-Tai Lai, Chao-Feng Tseng, "Laplacian-based H.264 intra-prediction mode decision," Communications and Networking in China (CHINACOM), 2012 7th International ICST Conference on , vol., no., pp.638,641, 8-10 Aug. 2012.

[11] Sourabh Rungta, Kshitij Verma and Anupam Shukla, "A Fast Mode Selection Algorithm Using Texture Analysis for H.264/AVC," IJCSI International of computer Sciences Issues, Vol. 7, Issue 4,No 9, July 2010.

[12] Mohammed Golam Sarwer and Q. M. Jonathan Wu, "Improved Intra Prediction of H.264/AVC," Effective Video Coding for Multimedia Applications, Sudhakar Radhakrishnan (Ed.), ISBN: 978-953-307-177-0, InTech (2011).

[13] Yeong-Il Jeon; Chan-Hee Han; Si-Woong Lee; Hyun-Soo Kang, "Fast Intra Mode Decision Algorithm Using Directional Gradients for H.264," Image and Signal Processing, 2009. CISP '09. 2nd International Congress on , vol., no., pp.1,4, 17-19 Oct. 2009.

[14] A.Elyousfi, A.Tamtaoui and E.Bouyakhf, "Fast Intra Prediction Algorithm for H.264/AVC Based on Quadratic and Gradient Model," World Academy of Science, Engineering and Technology 63, 2010.

[15] Yuri V. Ivanov and C. J. Bleakley. 2010. Real-time H.264 video encoding in software with fast mode decision and dynamic complexity control. ACM Trans. Multimedia Comput. Commun. Appl.6, 1, Article 5 (February 2010), 21 pages.

[16] De Cock, Jan, Stijn Notebaert, Peter Lambert, and Rik Van de Walle. 2006. "Hardware/software Co-design for H.264/AVC Intra Frame Encoding." In EUROMEDIA '2006, ed. E Tzafestas, 56–60. Ghent, Belgium: EUROSIS.

[17] [17] Colenbrander, R.R.; Damstra, A.S.; Korevaar, C.W.; Verhaar, C.A.; Molderink, A.; , "Co-design and Implementation of the H.264/AVC Motion Estimation Algorithm Using Co-simulation," Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference, p.210-215, 3-5 Sept. 2008.

[18] Chih-Hung Kuo, Li-Chuan Chang, Kuan-Wei Fan, Bin-Da Liu, "Hardware/Software Codesign of a Low-Cost Rate Control Scheme for H.264/AVC," IEEE Transactions on Circuits and Systems for Video Technology Volume 20 Issue 2, February 2010.

[19] Yahya Jan, Lech Jozwiak, "CABAC Accelerator Architectures for Video Compression in Future Multimedia: A Survey," Embedded Computer Systems: Architectures, Modeling, and Simulation, Lecture Notes in Computer Science Volume 5657, 2009, pp 24-35.

[20] Moez Kthiri, Hassen Loukil, Ahmed Ben Atitallah, Patrice Kadionik, Dominique Dallet, Nouri Masmoudi "FPGA architecture of the LDPS Motion Estimation for H.264/AVC Video Coding," Journal of Signal Processing Systems , August 2012, Volume 68, Issue 2, pp 273-285.

[21] Dias, T.; Roma, N.; Sousa, L., "H.264/AVC framework for multi-core embedded video encoders," System on Chip (SoC), 2010 International Symposium on , vol., no., pp.89,92, 29-30 Sept. 2010

[22] M. Bariani, P. Lambruschini, and M. Raggio, "An Efficient Multi-Core SIMD Implementation for H.264/AVC Encoder," VLSI Design, vol. 2012, Article ID 413747, 14 pages, 2012.

[23] Shenggang Chen; Shuming Chen; Huitao Gu; Hu Chen; Yaming Yin; Xiaowen Chen; Shuwei Sun; Sheng Liu; Yaohua Wang, "Mapping of H.264/AVC Encoder on a Hierarchical Chip Multicore DSP Platform," High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on , vol., no., pp.465,470, 1-3 Sept. 2010

[24] Imen Werda, Haithem Chaouch, Amine Samet, Mohamed Ali Ben Ayed and Nouri Masmoudi, "Optimal DSP-Based Motion Estimation Tools Implementation For H.264/AVC Baseline Encoder," IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.5, May 2007.

[25] Jan-Willem van de Waerdt, Gerrit A. Slavenburg, Jean-Paul van Itegem, and Stamatis Vassiliadis. 2005. Motion estimation performance of the TM3270 processor. In Proceedings of the 2005 ACM symposium on Applied computing (SAC '05), Lorie M. Liebrock (Ed.).

[26] R. Vani, M. Sangeetha, "Survey on H.264 Standard," Advances in Computer Science and Information Technology, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, Volume 86, 2012, pp 397-410.

[27] Daw-Tung Lin, Chung-Yu Yang, "H.264/AVC Video Encoder Realization and Acceleration on TI DM642 DSP," Advances in Image and Video Technology Lecture Notes in Computer Science Springer Berlin Heidelberg, Volume 5414, 2009, pp 910-920.

[28] Hung-Chih Lin; Yu-Jen Wang; Kai-Ting Cheng; Shang-Yu Yeh; Wei-Nien Chen; Chia-Yang Tsai; Tian-Sheuan Chang; Hsueh-Ming Hang; , "Algorithms and DSP implementation of H.264/AVC," Design Automation, 2006. Asia and South Pacific Conference on , vol., no., pp.8 pp., 24-27 Jan. 2006.

[29] Wonchul Lee, Hyojin Choi, Wonyong Sung "Algorithm and Software Optimization of Variable Block Size Motion Estimation for H.264/AVC on a VLIW–SIMD DSP," Journal of Signal Processing Systems June 2008, Volume 51, Issue 3, pp 289-302.

[30] Werda, I; Kossentini, F; Ben Ayed, M.-A; Nouri Massmoudi; "Analysis and Optimization of UB Video's H.264 Baseline Encoder Implementation on Texas Instruments' TMS320DM642 DSP," Image Processing, 2006 IEEE International Conference on , vol., no., pp.3277-3280, 8-11 Oct. 2006.

[31] Mohammadnia, M.R.; Taheri, H.; Motamedi, S.A.; "Implementation and Optimization of Real-Time H.264/AVC Main Profile Encoder on DM648 DSP," Signal Acquisition and Processing, 2009. ICSAP 2009. International Conference on , vol., no., pp.48-52, 3-5 April 2009.

[32] Damak, T; Werda, I; Samet, A; Masmoudi, N; "DSP CAVLC implementation and optimization for H.264/AVC baseline encoder," Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on, vol., no., pp.45-48, Aug. 31 2008-Sept. 3 2008.

[33] Zhengming Li; Qiuyan Xing; Xiaoyong Zhu;, "H.264 video encoder implementation and optimization based on DM642 DSP," Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on , vol., no., pp.891-894, 6-8 April 2008.

[34] TMS320C6472 datasheet
http://www.ti.com/lit/ds/sprs612g/sprs612g.pdf.

[35] TMS320C6472/TMS320TCI648x DSP Enhanced DMA (EDMA3) Controller. http://www.ti.com/lit/ug/spru727e/spru727e.pdf

[36] TMS320C6472 Chip Support Library API reference Guide.
http://software-dl.ti.com/sdoemb/sdoemb_public_sw/csl/CSL_C6472/latest/index_FDS.html

[37] TCP/IP socket programming.
http://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/

[38] TI Network Developer's Kit (NDK) v2.21 User's Guide, http://www.ti.com/lit/ug/spru523h/spru523h.pdf

[39] Open source computer vision library (OpenCv library) http://opencv.org/